57. ročník Matematické olympiády – 2007/2008

Úlohy ústředního kola kategorie P – 1. soutěžní den

Na řešení úloh máte 4,5 hodiny čistého času.

Řešení každého příkladu musí obsahovat:

- **Popis řešení,** to znamená slovní popis použitého algoritmu, argumenty zdůvodňující jeho správnost (případně důkaz správnosti algoritmu), diskusi o efektivitě vašeho řešení (časová a paměťová složitost). Slovní popis řešení musí být jasný a srozumitelný i bez nahlédnutí do samotného zápisu algoritmu (do programu). Není vhodné odkazovat se na Vaše řešení předchozích kol, opravovatelé je nemají k dispozici; na autorská řešení se odkazovat můžete.
- **Program.** V úlohách **P-III-1** a **P-III-2** je třeba uvést dostatečně podrobný zápis algoritmu, např. ve tvaru zdrojového textu nejdůležitějších částí programu v jazyce Pascal nebo C. Ze zápisu můžete vynechat jednoduché operace jako vstupy, výstupy, implementaci jednoduchých matematických vztahů apod. V řešení úlohy **P-III-3** je nutnou součástí řešení přesný popis příslušných překládacích strojů.

Hodnotí se nejen správnost programu, ale také kvalita popisu řešení a efektivita zvoleného algoritmu.

P-III-1 Karpaty

Už zanedlouho si splníte svůj velký cestovatelský sen a vydáte se na vytouženou pouť přes celé Vnější Karpaty. Vše máte dobře naplánováno – začnete v Beskydech a budete pokračovat přes Slovensko, Polsko a Ukrajinu až do Rumunska, odkud už se chcete vrátit skrze civilizaci. Cestou máte v úmyslu vystoupit na co nejvíce hor, kolem kterých pojedete, ale jak se znáte, tak jakmile vystoupíte na nějakou horu, už se Vám nebude chtít na žádnou stejně vysokou nebo dokonce nižší.

V rámci pečlivého plánování jste si vypsali výšky všech hor v tom pořadí, v jakém kolem nich budete projíždět a přemýšlíte, na které z hor budete chtít vystoupit, abyste vystoupili během své cesty na co největší počet hor. Po krátkém zamyšlení jste si uvědomili, že těch nejdelších posloupností hor takových, že jejich výšky striktně rostou, může být i více. Místo výčtu všech možností byste si raději u každé hory poznačili, zda ji navštívíte určitě (tedy pokud je v každé nejdelší posloupnosti hor, kde výšky hor rostou), zda ji můžete navštívit, nebo ji určitě nenavštívíte, pokud chcete vystoupat na co největší počet hor.

Soutěžní úloha

Pro zadanou posloupnost celých kladných čísel určete, která ze zadaných čísel leží v každé nejdelší rostoucí podposloupnosti, v nějaké nejdelší rostoucí podposloupnosti a v žádné nejdelší rostoucí podposloupnosti.

Formát vstupu

Vstup je tvořen dvěma řádky. Na prvním řádku je jedno celé číslo N, které určuje počet hor na vaší cestě. Druhý řádek pak obsahuje N celých kladných čísel $v_1 \dots v_n$, která udávají výšku hor.

Formát výstupu

Výstup je tvořen jedním řádkem s N slovy oddělenými jednou mezerou. Slova ve výstupu jsou '`musím`', '`mohu`' a '`nemohu`' podle toho, zda na danou horu musíte, můžete nebo nemůžete vystoupat, pokud chcete během své cesty zdolat co největší počet hor tak, aby jejich výšky postupně rostly.

Příklad

Vstup:

```
9
3 2 1 4 5 6 1 8 7
```

Výstup:

```
mohu mohu mohu musím musím musím nemohu mohu mohu
```

Nejdelší posloupnost rostoucích hor zde má délku 5 a z první trojice a poslední dvojice vždy obsahuje právě jeden prvek.

P-III-2 Velechrám

Oslavy 75. narozenin královny Šeherezády se blíží a v paláci se to jen hemží. Vyměňují se tapisérie, pokládají se nové koberce a přemalovávají se obrazy. Radní se rozhodli, že královnu překvapí novou duhovou výzdobou velechrámu. Chtějí nechat natřít sloupy v chrámu různými barvami, aby z toho jen oči přecházely. Aby byl výsledný dojem co nejlepší, je třeba, aby v žádné řadě sloupů neměly dva sloupy stejnou barvu. Královská pokladna však není bezedná, a proto je třeba nakoupit co nejméně druhů barev. Proto si vás zavolali na pomoc.

Soutěžní úloha

Velechrám si lze představit jako šachovnici o velikosti $N \times N$. Na čtvercových polích stojí M sloupů, tj. ne na všech polích stojí sloup. Žádné dva sloupy nestojí na stejném poli. Vaším úkolem je obarvit tyto sloupy co nejmenším počtem barev K tak, aby žádné dva sloupy ve stejném řádku nebo sloupci neměly stejnou barvu.

Formát vstupu

Na prvním řádku dostanete dvě přirozená čísla N a M, velikost šachovnice $1 \leq N \leq 500$ a počet sloupů $0 \leq M \leq N^2$.

Následuje M řádků popisujících polohu sloupů. Na i-tém z nich najdete dvě celá čísla R_i a S_i, řádek $1 \leq R_i \leq N$ a sloupec $1 \leq S_i \leq N$ pole, na kterém stojí i-tý sloup.

Formát výstupu

První řádek výstupu obsahuje minimální počet barev K. Na následujících M řádků vypište barvy sloupů. Na i-tý z těchto řádků napište číslo barvy B_i,

$1 \leq B_i \leq K$, kterou má být obarven i-tý sloup. Optimálních obarvení bude zřejmě více (barvy lze například mezi sebou libovolně permutovat), vypište proto libovolné z nich.

Příklady

Vstup:	*Výstup:*
`3 4`	`2`
`1 1`	`1`
`1 3`	`2`
`3 3`	`1`
`3 1`	`2`

Sloupy leží ve vrcholech čtverce. Protilehlé sloupy dostanou stejnou barvu.

Vstup:	*Výstup:*
`4 8`	`3`
`1 1`	`1`
`1 3`	`2`
`1 4`	`3`
`2 1`	`2`
`3 3`	`3`
`4 1`	`3`
`4 2`	`2`
`4 3`	`1`

P-III-3 Překládací stroje

Studijní text, který je stejný jako v domácím kole, následuje po zadání soutěžní úlohy.

Uvažme následující množiny A, B, C a D řetězců. Množina A obsahuje všechny dobře uzávorkované řetězce znaků '`(`', '`)`', '`[`', '`]`', '`{`' a '`}`'. Množinu A lze rekurzivně nadefinovat následujícím způsobem: A obsahuje prázdný řetězec ε a všechny řetězce tvaru `(`$\alpha_1 \ldots \alpha_k$`)`, `[`$\alpha_1 \ldots \alpha_k$`]` a `{`$\alpha_1 \ldots \alpha_k$`}`, kde $\alpha_1, \ldots, \alpha_k$ jsou nějaké řetězce v A již obsažené. Množina A tedy obsahuje řetězce `(([{}]))` a `()()(){}`, ale neobsahuje ani jeden z řetězců `([)]` a `())(`.

Množiny B a C jsou množiny řetězců nad dvojpísmennou abecedou $\{a, b\}$. Množina B obsahuje řetězce nejprve tvořené $n \geq 0$ písmeny a a pak alespoň $2n$ písmeny b. Množina C obsahuje ty řetězce, které jsou tvořeny stejným počtem písmen a a b. Např. $aabbbbb \in B$ a $abba \in C$, ale $aaabbbb \notin B$, $bbba \notin B$ a $ababb \notin C$.

Množina D je tvořena těmi řetězci, které nejprve obsahují $n \geq 0$ písmen a, pak n písmen b a nakonec n písmen c. Tedy, $aaabbbccc \in D$, ale $aababbbccc \notin D$ a $bbbaaaccc \notin D$.

Soutěžní úloha

a) *(2 body)* Navrhněte překládací stroj, který přeloží množinu A na množinu B.

b) *(3 body)* Nalezněte množinu řetězců E nad abecedou $\{a, b\}$ a překládací stroje M_1 a M_2 takové, že stroj M_1 přeloží množinu E na množinu A a stroj M_2 přeloží množinu E na množinu D.

c) *(5 bodů)* Sestrojte překládací stroj, který přeloží množinu A na množinu C. Nezapomeňte, že nutnou součástí řešení je i důkaz, že vámi navržený překládací stroj přeloží množinu A právě na množinu C.

Studijní text

Překládací stroj přijímá na vstupu řetězec znaků. Tento řetězec postupně čte a podle předem zvolené soustavy pravidel (tedy podle svého programu) občas nějaké znaky zapíše na výstup. Když stroj zpracuje celý vstupní řetězec a úspěšně ukončí svůj výpočet, vezmeme řetězec znaků zapsaný na výstup a nazveme ho *překladem* vstupního řetězce.

Výpočet stroje nemusí být jednoznačně určen. Jinými slovy, soustava pravidel může někdy stroji umožnit, aby se rozhodl o dalším postupu výpočtu. V takovém případě se může stát, že některý řetězec bude mít více různých překladů.

Naopak, může se stát, že v určité situací se podle daných pravidel nebude moci v překladu pokračovat vůbec. V takovém případě se může stát, že některý řetězec nebude mít vůbec žádný překlad.

Formálnější definice překládacího stroje

Každý překládací stroj pracuje nad nějakou předem zvolenou konečnou množinou znaků. Tuto množinu znaků budeme nazývat *abeceda* a značit Σ. V soutěžních úlohách bude vždy Σ známa ze zadání úlohy. Abeceda nebude nikdy obsahovat znak \$, ten budeme používat k označení konce vstupního řetězce.

Stroj si může během překladu řetězce pamatovat informaci konečné velikosti. Formálně tuto skutečnost definujeme tak, že stroj se v každém okamžiku překladu nachází v jednom z *konečně mnoha* stavů. Nutnou součástí programu překládacího stroje je tedy nějaká konečná *množina stavů*, v nichž se stroj může nacházet. Tuto množinu označíme K. Kromě samotné množiny stavů je také třeba uvést, ve kterém stavu se stroj nachází na začátku každého překladu. Tento stav nazveme *počáteční stav*.

Program stroje se skládá z konečného počtu překladových pravidel. Každé pravidlo má tvar čtveřice (p, u, v, q), kde $p, q \in K$ jsou nějaké dva stavy a u, v jsou nějaké dva řetězce znaků z abecedy Σ.

Stavy p a q mohou být i stejné. Řetězec u může být tvořen jediným znakem \$. Řetězce u a v mohou být i stejné. Některý z těchto řetězců může být případně prázdný. Aby se program lépe četl, budeme místo prázdného řetězce psát symbol ε.

Překladové pravidlo má následující význam: „Když je stroj právě ve stavu p a dosud nepřečtená část vstupu začíná řetězcem u, může stroj tento řetězec ze vstupu přečíst, na výstup zapsat řetězec v a změnit svůj stav na q.“ Všimněte si, že pravidlo tvaru (p, ε, v, q) můžeme použít vždy, když se stroj nachází ve stavu p, bez ohledu na to, jaké znaky ještě zůstávají na vstupu.

Ještě potřebujeme stanovit, kdy překlad úspěšně skončil. V první řadě budeme požadovat, aby překládací stroj přečetl celý vstupný řetězec. Kromě toho umožníme stroji „odpovědět“, zda se mu překlad podařil nebo ne. To zařídíme tak, že některé stavy stroje označíme jako *koncové stavy*. Množinu všech koncových stavů budeme značit F.

Formální definice překládacího stroje

Překládací stroj je uspořádaná pětice (K, Σ, P, q_0, F), kde Σ a K jsou *konečné* množiny, $q_0 \in K$, $F \subseteq K$ a P je *konečná* množina překladových pravidel popsaných výše. Přesněji, nechť Σ^* je množina všech řetězců tvořených znaky ze Σ, potom P je *konečná* podmnožina množiny $K \times (\Sigma^* \cup \{\$\}) \times \Sigma^* \times K$.

(Pro každé $q \in K$ budeme množinu pravidel, jejichž první složkou je q, nazývat „překladová pravidla ze stavu q“.)

Chceme-li definovat konkrétní překládací stroj, musíme uvést všech pět výše uvedených objektů.

Když už máme definován konkrétní stroj $A = (K, \Sigma, P, q_0, F)$, můžeme určit, jak tento stroj překládá konkrétní řetězec. Uvedeme nejprve formální definici a potom ji slovně vysvětlíme.

Množina *platných překladů* řetězce u překládacím strojem A je:

$$A(u) = \Big\{ \quad v \quad \Big| \quad \begin{array}{l} \exists n \geq 0 \; \exists (p_1, u_1, v_1, r_1), \ldots, (p_n, u_n, v_n, r_n) \in P : \\ (\forall i \in \{1, \ldots, n-1\} : r_i = p_{i+1}) \;\wedge\; p_1 = q_0 \;\wedge\; r_n \in F \;\wedge \\ \quad \wedge \;\; \exists k \geq 0 : u_1 u_2 \ldots u_n = u \underbrace{\$ \ldots \$}_{k} \wedge v_1 v_2 \ldots v_n = v \end{array} \quad \Big\}.$$

Definice stanoví, kdy je řetězec v překladem řetězce u. Vysvětlíme si slovně význam jednotlivých řádků definice:

- První řádek říká, že aby se dalo u přeložit na v, musí existovat nějaká posloupnost překladových pravidel, kterou při tomto překladu použijeme. Další dva řádky popisují, jak tato posloupnost musí vypadat.
- Druhý řádek zabezpečuje, aby stavy v použitých pravidlech byly správné: První pravidlo musí být pravidlem z počátečního stavu, každé další pravidlo musí být pravidlem z toho stavu, do něhož se stroj dostal použitím předcházejícího pravidla.

  Navíc stav, v němž se bude stroj nacházet po skončení výpočtu, musí být koncový.
- Poslední řádek popisuje řetězce, které stroj při použití dotyčných překladových pravidel čte a zapisuje.

  Řetězec, který při použití těchto pravidel stroj přečte ze vstupu, musí být skutečně zadaným řetězcem u, případné může být zprava doplněn vhodným počtem znaků \$.

  Řetězec, který stroj zapíše na výstup, musí být přesně řetězcem v.

K čemu budeme používat překládací stroje?

Překládací stroje nám budou sloužit k získání překladu jedné množiny řetězců na jinou množinu řetězců. Jestliže A je překládací stroj a $M \subseteq \Sigma^*$ nějaká množina řetězců, potom překlad množiny M strojem A je množina

$$A(M) = \bigcup_{u \in M} A(u).$$

Jinými slovy, výslednou množinu $A(M)$ sestrojíme tak, že vezmeme všechny řetězce z M a pro každý z nich přidáme do $A(M)$ všechny jeho platné překlady.

Příklad 1

Mějme abecedu $\Sigma = \{0, \ldots, 9\}$. Nechť M je množina všech řetězců, které představují zápisy kladných celých čísel v desítkové soustavě. Sestrojíme překládací stroj A, pro který bude platit, že překladem této množiny M bude množina zápisů všech kladných celých čísel, která jsou dělitelná třemi.

Řešení

Nejjednodušší bude prostě vybrat z M ta čísla, která jsou dělitelná třemi. Náš překládací stroj bude kopírovat cifry ze vstupu na výstup, přičemž si bude pomocí stavu pamatovat, jaký zbytek po dělení třemi dává dosud přečtené (a zapsané) číslo. Nachází-li se po dočtení vstupu ve stavu odpovídajícím zbytku 0, přejde do koncového stavu.

Formálně A bude pětice $(K, \Sigma, P, 0, F)$, kde $K = \{0, 1, 2, end\}$, $F = \{end\}$ a překladová pravidla vypadají následovně:

$$P = \Big\{(x, y, y, z) \;\Big|\; x \in \{0, 1, 2\} \;\wedge\; y \in \Sigma \;\wedge\; z = (10x + y) \bmod 3\Big\} \;\cup\; \Big\{(0, \$, \varepsilon, end)\Big\}.$$

Příklad 2

Mějme abecedu $\Sigma = \{a, e, i, \bullet, \text{━}\}$. Sestrojíme překládací stroj B, pro který bude platit, že překladem libovolné množiny M, která obsahuje pouze řetězce z písmen a, e a i, bude množina stejných řetězců zapsaných v morseovce (bez oddělovačů mezi znaky). Zápisy našich písmen v morseovce vypadají následovně: a je $\bullet$ ━, e je $\bullet$ a i je $\bullet\bullet$.

Například množinu $M = \{ae, eea, ia\}$ by náš stroj měl přeložit na množinu $\{\bullet \text{━} \bullet, \bullet \bullet \bullet \text{━}\}$. (Všimněte si, že řetězce eea a ia mají v morseovce bez oddělovačů stejný zápis.)

Řešení

Překládací stroj B bude číst vstupní řetězec po znacích a vždy zapíše na výstup kód přečteného znaku.

Formálně B bude pětice $(K, \Sigma, P, \heartsuit, F)$, kde $K = \{\heartsuit\}$, $F = \{\heartsuit\}$ a překladová pravidla vypadají takto:

$$P = \{(\heartsuit, a, \bullet \text{━}, \heartsuit),\ (\heartsuit, e, \bullet, \heartsuit),\ (\heartsuit, i, \bullet\bullet, \heartsuit)\}.$$

Všimněte si, že nepotřebujeme nijak zvlášť kontrolovat, zda jsme na konci vstupu. Během celého překladu je totiž stroj v koncovém stavu, takže jakmile přečte poslední znak ze vstupu, bude vytvořený překlad platný.

Příklad 3

Mějme abecedu $\Sigma = \{a, e, i, \bullet, \text{━}\}$. Sestrojíme překládací stroj C, pro který bude platit, že překladem libovolné množiny M, která obsahuje pouze řetězce tvořené znaky $\bullet$ a ━, bude množina *všech* řetězců z písmen a, e a i, jejichž zápisy v morseovce (bez oddělovačů mezi znaky) jsou obsaženy v množině M. Například množinu $M = \{\bullet\text{━}\bullet, \bullet\bullet\bullet\text{━}\}$ by náš stroj měl přeložit na $\{ae, eea, ia\}$.

Řešení

Našemu překládacímu stroji dáme možnost rozhodnout se v každém okamžiku překladu, že bude číst kód nějakého písmena a zapíše na výstup toto písmeno. Potom každé možnosti, jak lze rozdělit vstupní řetězec na kódy písmen, bude odpovídat jeden platný překlad.

Formálně C bude pětice $(K, \Sigma, P, \diamondsuit, F)$, kde $K = \{\diamondsuit\}$, $F = \{\diamondsuit\}$ a překladová pravidla vypadají následovně:

$$P = \{(\diamondsuit, \bullet\text{━}, a, \diamondsuit),\ (\diamondsuit, \bullet, e, \diamondsuit),\ (\diamondsuit, \bullet\bullet, i, \diamondsuit)\}.$$

Ukážeme si, jak mohl probíhat překlad řetězců z výše uvedené množiny M. Existují tyto tři možnosti:

$(\diamondsuit, \bullet\text{━}, a, \diamondsuit), (\diamondsuit, \bullet, e, \diamondsuit)$
$(\diamondsuit, \bullet\bullet, i, \diamondsuit), (\diamondsuit, \bullet\text{━}, a, \diamondsuit)$
$(\diamondsuit, \bullet, e, \diamondsuit), (\diamondsuit, \bullet, e, \diamondsuit), (\diamondsuit, \bullet\text{━}, a, \diamondsuit)$

Kdybychom zkusili pro libovolný vstupní řetězec z M použít překladová pravidla v jiném pořadí – např. pro vstup $\bullet\bullet\bullet\text{━}$ použít třikrát pravidlo $(\diamondsuit, \bullet, e, \diamondsuit)$ – nepodaří se nám dočíst vstupní řetězec až do konce.

Příklad 4

Mějme abecedu $\Sigma = \{a, b, c\}$. Nechť množina X obsahuje právě všechny řetězce, v nichž je obsažen stejný počet znaků a a b. Tedy například $abbccac \in X$, ale $cbaa \notin X$.

Nechť množina Y obsahuje právě všechny řetězce, které neobsahují žádné a, neobsahují podřetězec bc, a písmen c je dvakrát více než písmen b. Tedy například $ccccbb \in Y$, ale $cccbcb \notin Y$ a $acacba \notin Y$.

Sestrojíme překládací stroj D, který přeloží X na Y.

Řešení

Budeme překládat jenom některé vhodné řetězce z množiny X. Budou to ty řetězce, které neobsahují žádné c a všechna a v nich předcházejí všem znakům b. Takovýto řetězec přeložíme tak, že nejprve každé a přepíšeme na cc, a potom zkopírujeme na výstup všechna b. Tedy například překladem slova $aabb$ bude slovo $ccccbb$.

Formálně D bude pětice $(K, \Sigma, P, \textbf{čti-a}, F)$, kde $K = \{\textbf{čti-a}, \textbf{čti-b}\}$, $F = \{\textbf{čti-b}\}$ a překladová pravidla vypadají takto:

$$P = \{(\textbf{čti-a}, a, cc, \textbf{čti-a}),\ (\textbf{čti-a}, \varepsilon, \varepsilon, \textbf{čti-b}),\ (\textbf{čti-b}, b, b, \textbf{čti-b})\}.$$

Proč tento překládací stroj funguje? Když vstupní řetězec obsahuje nějaké písmeno c, při jeho překládání se u prvního výskytu c náš stroj zastaví. Proto takové řetězce nemají žádný platný překlad. Podobně nemají platný překlad řetězce, v nichž není dodrženo pořadí písmen a a b. Po přečtení nějaké posloupnosti písmen a přejde stroj pomocí druhého pravidla do stavu **čti-b**, a pokud se poté ještě objeví na vstupu a, stroj se zastaví.

Platné překlady tedy existují skutečně pouze pro slova výše popsaného tvaru. Je zjevné, že překladem každého z nich získáme nějaký řetězec z Y, takže $D(X) \subseteq Y$. Naopak, vybereme-li si libovolné slovo z Y, snadno najdeme slovo z X, které se na něj přeloží. Proto také $Y \subseteq D(X)$, takže $Y = D(X)$.